Attorney Docket No.
ADAPP243

PATENT APPLICATION

INTEGRATED CIRCUIT VERIFICATION SCHEME

INVENTORS: (1) Purna Mohanty
2191 Esperanca Ave
Santa Clara, CA 95054
Citizen of India

(2) Sivam Thangavelu
2888 Glen Hawkins Ct.
San Jose, CA 95148
Citizen of India

(3) Vivek Jambhekar
870 East El Camino Real Apt. #42
Mountain View, CA 94040
Citizen of India

ASSIGNEE: Adaptec, Inc.
691 S. Milpitas Boulevard
Milpitas, CA 95035

MARTINE & PENILLA, LLP
710 Lakeway Dr., Suite 170
Sunnyvale, California 94085
Telephone (408) 749-6900

INTEGRATED CIRCUIT VERIFICATION SCHEME

by Inventors:

PURNA MOHANTY, SIVAM THANGAVELU AND VIVEK JAMBHEKAR

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application is related to U.S. Patent Application No. _____ (Attorney Docket No. ADAPP244), filed on the same day as the instant application and entitled "METHOD AND APPARATUS FOR ACCELERATING TEST CASE DEVELOPMENT." This application is hereby incorporated by reference in its entirety.

BACKGROUND OF THE INVENTION

1. Field of the Invention

[0002] The present invention relates generally to chip design schemes, and more specifically to application specific integrated circuit (ASIC) validation schemes that reduce the validation time without sacrificing quality.

2. Description of the Related Art

[0003] Development timelines for integrated circuits (ICs) are constantly being evaluated in order to identify efficiencies for reducing the development timeline in order to make the IC available to the market as fast as possible. At the design stage for an integrated circuit, a circuit design is typically represented as circuit data in a hardware description language (HDL). From the HDL, the circuit design can be mapped (through suitable processing) into an actual hardware design for fabrication in an integrated circuit. Standards exist for HDLs, e.g., Verilog standards.

[0004] The testing of circuit designs prior to fabrication in an integrated circuit is referred to in the art as verification and validation, and represents an important step in the design process for an integrated circuit. With the complexity of circuits increasing continuously, it is impossible to guarantee the proper operation of a design without undergoing an extensive prior verification and validation of the design. The HDL-based design methodology requires the user to describe the behavior of a system which may then be simulated (a process referred to as Register Transfer Level (RTL) simulation) to determine whether it will function as desired. The design is then synthesized to create a logical netlist that can be implemented within a particular Programmable Logic Device (PLD) or a particular PLD vendor's device group. Assuming that neither the synthesis or the design implementation processes alter the behavior of the design, RTL simulation provides the designer with insight into the functionality and behavior of the implemented device before the costly and time consuming synthesis and design implementation processes take place. If accurate, RTL simulation represents a quick and efficient means of validating a design's (and ultimately, a programmed device's) behavior.

[0005] During the validation process, complete testcases are created as a macrolevel Verilog task. The test cases are compiled along with the testbench and the design. And the simulation is then run. Once the simulation is run it is determined whether the goal of the test case is met. If the goal of the test case is not met, then the RTL is checked. If the RTL check is satisfactory, then the tasks defining the test case are adjusted. It should be appreciated that this process is iterative until the desired tasks for the testcase are determined. At each iteration, the newly defined testcase must be compiled, which consumes CPU resources of the computer system running a commercially available

simulation program, e.g., a verilog based simulation program, as well as seat licenses for the simulation program.

[0006] As licenses are generally structured to allow a user to perform a compilation or a simulation, and not both at the same time, the user will either have to wait for a license to be available if a simulation is running, or use an additional seat license for the compilation. Thus, an organization will generally have to purchase additional seat licenses to avoid the downtime. Furthermore, the compilations are costly in terms of the design engineer's time and are a bottleneck in the validation process.

[0007] In light of the foregoing, it is desired to implement a scheme to validate a design for an integrated circuit in a more efficient manner thereby reducing overhead associated with machine/license resources due to the constant re-compilations.

SUMMARY OF THE INVENTION

[0008] Broadly speaking, the present invention fills these needs by providing a verification scheme where the task files are written as text files and are associated with a precompiled executable in order to eliminate the need for re-compiling the test case when tasks are adjusted. The present invention can be implemented in numerous ways, including as an apparatus, a system, a device, a computer readable media or a method. Several embodiments of the present invention are described below.

[0009] In one embodiment, a method for verifying an integrated circuit design is provided. The method initiates with defining tasks. The defined tasks are formatted as text files. Then, a test case is identified. The test case is defined by the tasks. Next, each of the tasks are correlated with a compiled task formatted in a hardware description language. Then, the compiled tasks are executed to simulate a behavior for the integrated circuit design to determine if a goal has been achieved. If the goal has not been achieved, the method includes adjusting the tasks of the test case and repeating the execution of the test case to simulate the behavior for the integrated circuit design.

[0010] In another embodiment, a method for minimizing compilation time of a test case during development testing of an integrated circuit is provided. The method initiates with identifying a test case. The test case is associated with the tasks and the tasks are written as text files. Then, a file associated with the test case is generated. Next, a sequence of the tasks of the file is determined. Then, hardware description language (HDL) tasks, associated with the tasks of the file according to the sequence, are identified. Next, a simulation of an integrated circuit is performed through the HDL tasks.

[0011] In still yet another embodiment, a computer readable media having program instructions for avoiding re-compilation of a test case during development testing of an integrated circuit is provided. The computer readable media includes program instructions for identifying a test case. The test case is associated with tasks written as text files. Program instructions for generating a file associated with the test case are provided. Program instructions for determining a sequence of the tasks of the file and program instructions for identifying hardware description language (HDL) tasks associated with the tasks of the file according to the sequence are included. Program instructions for performing a simulation of an integrated circuit with the HDL tasks are also provided.

[0012] In another embodiment, a computer system for validating an integrated circuit design is provided. The computer system includes a processor and a memory. A storage device is included. The storage device is configured to store compiled tasks written in a hardware description language (HDL) for testing the functional characteristics of an integrated circuit design. Each of the compiled tasks is associated with a corresponding text file. A bus enabling communication between the processor, the memory and the storage device is provided.

[0013] Other aspects and advantages of the invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrating by way of example the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] The accompanying drawings, which are incorporated in and constitute part of this specification, illustrate exemplary embodiments of the invention and together with the description serve to explain the principles of the invention.

[0015] Figure 1 is a high level simplified schematic diagram of a system configured to minimize the validation time of an integrated circuit in accordance with one embodiment of the invention.

[0016] Figure 2 is a simplified block diagram illustrating the test case execution flow in accordance with one embodiment of the invention.

[0017] Figure 3 is a simplified schematic diagram of a system which enables the efficient validation of a integrated circuit design in accordance with one embodiment of the invention.

[0018] Figure 4 is a flow chart diagram illustrating the method operations for efficiently validating a chip design in accordance with one embodiment of the invention.

[0019] Figure 5 is a flowchart diagram representing the method operations for minimizing compilation time of a test case during development testing of an integrated circuit in accordance with one embodiment of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0020] An invention is described for a system, device and method for efficiently verifying a design of an integrated circuit (IC). It will be obvious, however, to one skilled in the art, that the present invention may be practiced without some or all of these specific details. In other instances, well known process operations have not been described in detail in order not to unnecessarily obscure the present invention. The term about as used to herein refers to +/- 10% of the referenced value.

[0021] The embodiments of the present invention provide a method and system for performing the verification/validation of an IC design prior to implementing the design onto silicon. The verification/validation scheme described herein defines tasks which are applied to a model of the design to elicit a response. The tasks are text files which are correlated to a precompiled task formatted in a hardware description language (HDL). A translation layer associates the tasks formatted as text files with the precompiled tasks. Through the use of the tasks formatted as text files and the precompiled tasks, adjustments made to test cases during the design phase are implemented without the need to recompile the test cases.

[0022] Figure 1 is a high level simplified schematic diagram of a system configured to minimize the validation time of an integrated circuit in accordance with one embodiment of the invention. Here, a list of tasks in block 100 are stored in storage media 102. It should be appreciated that the tasks are stored as compiled tasks 104 as will be explained in more detail below. From the list of tasks 100, a number of test cases (TC) may be generated. For example, TC_1 106-1 includes task 0 and task 1 while TC_2 106 -2 includes task 3, task 1 and task 4. Likewise, TC_n 106-n includes task 0, through task n. In one

embodiment, the compiled tasks, also referred to as precompiled tasks, are compiled verilog files, e.g., a library of compiled files. Thus, when it is time to execute test case 1 for the validation of a chip design, task 0 and task 1, which make up test case 1, are executed in block 108-1. It should be appreciated that where necessary, translation layer 110-1 will translate the respective tasks in order for them to be executed. That is, the tasks are written as text files and associated with an executable verilog file. Thus, the translation layer associates each of the tasks of the text file with a corresponding verilog file. Similarly, for the execution of test case n, the respective tasks are executed in block 108-2 and translated through translation layer 110-2.

[0023] Figure 2 is a simplified block diagram illustrating the test case execution flow in accordance with one embodiment of the invention. Block 114 includes macro task 1 through macro task n. Each of the macro tasks, i.e., macro task 1 114-1 through macro task n 114-n include groups of tasks. In one embodiment, the groups of tasks for each macro task are commonly used tasks, e.g., an initialization command. It should be appreciated that each of the macro tasks of block 114 are text files. Block 116 represents a defined test case. Here, a test case may include individual tasks and any group of tasks, i.e., macro tasks from block 114. The test case of block 116 is written as an ascii text file. The ascii text file of block 116 is then preprocessed in block 118. Here, a testinput.txt file checks for correct syntax and format. In one embodiment, a script that is capable of reading the text file ensures that the text file is in the proper syntax and generates a verified file. The generated file from block 118, which has been verified, is then processed by block 122. It should be appreciated that blocks 122 and 124 correspond to translation layer 110 with reference to Figure 1. In block 122 a generated file from the testinput.txt file of block 118 is analyzed in order to determine a sequence of tasks and

identify the tasks to stimulate a model of the chip design. Taskdispenser.c 122 reads each line of the testinput.txt file from block 118 and passes the task name and its arguments to taskdispenser.c of block 124. It should be appreciated that taskdispenser.c of block 122 generates files formatted in a C language.

[0024] Still referring to Figure 2, taskdispenser.v of block 124 then takes the task name and its arguments generated from block 122 and calls the appropriate tasks from tasklist.v in block 126. It should be appreciated that the taskdispenser.v file is a verilog file. Tasklist.v 126 is a precompiled list of verilog tasks in one embodiment of the invention. Thus, taskdispenser.v of block 124 correlates the task name from block 122 to the appropriate verilog file in tasklist.v of block 126. The correlated verilog file is then used to send stimuli to the test bench of block 128.

[0025] It will be apparent to one skilled in the art that the test bench model of block 128 of Figure 2 represents the surrounding environment or architecture of the model of the appropriate device under test (DUT) 130, which may be a model of an application specific to integrated circuit (ASIC). Thus, test bench block 128 imitates the behavior of the system in which the device being designed will operate. As the simulation runs, input stimulus are applied to DUT 130 and outputs are monitored. Once the simulation has completed, a simulation output file is generated and analyzed to ensure tester timing and format constraints have not been violated as well as ensuring the device functionally performed as expected. It should be appreciated that blocks 122, 124, 126, 128 and 130 represent a single run time case. As can be seen from the schematic diagram of Figure 2, there is no need to compile each time a task is adjusted or a test case is built. That is, there is one executable file sitting in one directory that is linked in order to provide the stimulation of the design of the integrated circuit.

[0026] Figure 3 is a simplified schematic diagram of a system which enables the efficient validation of a integrated circuit design in accordance with one embodiment of the invention. In one embodiment, the system of Figure 3 is a general computer system. System 140 includes processor 142, memory 144, storage media 148. Bus 146 enables communication between processor 142, memory 144 and storage media 148. Storage media 148 includes precompiled files 150. For example, precompiled files 150 represent the precompiled verilog tasks of tasklist.v 126 with reference to Figure 2. As discussed herein, the precompiled files enable the avoidance of having to compile a test case each time a change to a task of the test case is made.

[0027] It will be apparent to one skilled in the art that during verification/validation of an IC design, the test cases may be modified numerous times, e.g., the specific tasks of a test case may be substituted with another task in order to generate a specific response. Accordingly, the storage of the compiled verilog files are linked to corresponding text files defining the tasks. It should be appreciated that the invention is not limited to verilog files, as any suitable hardware description language (HDL) may be used to define the precompiled files. One skilled in the art will appreciate that storage media 148 may located externally or internal to the system of Figure 3. Furthermore, storage media 148 may be any suitable storage media for storing data. In addition, the storage media may be accessed through a network, either distributed or internal, so that the storage media may support a number of systems performing simulation operations associated with validation of a chip design.

[0028] Figure 4 is a flow chart diagram illustrating the method operations for efficiently validating a chip design in accordance with one embodiment of the invention. The method initiates with operation 160 where the test case goals are identified. Here, a list

of tasks may be assembled that stimulate a design of the integrated circuit in order to elicit a response captured through the simulation model. The method then advances to operation 164 where the tasks from the test case are used to simulate an activity. As discussed above, the tasks are correlated with a precompiled HDL file, e.g., verilog file, so that it is not necessary to compile each task. The method then proceeds to operation 166 where it is determined if the goal for the applicable test cases had been met. If the goal has been met, then the method proceeds to operation 168 where the chip design associated with that particular test case is passed.

[0029] If the goal has not been met in operation 166, the method moves to decision operation 170 where it is determined if the register transfer logic (RTL) is acceptable. If the RTL is not acceptable, the method advances to operation 172 where the RTL is fixed, i.e., repaired. The method then advances to operation 162 where the tasks from the repaired RTL are compiled. The compiled tasks are then used to simulate an activity of the chip design in operation 164 and proceeds as described above. If the RTL is acceptable in decision operation 170, then the method proceeds to operation 174 where the tasks are adjusted. For example, some tasks may be eliminated and replaced with other tasks, or some tasks may be added to the test case. The test case with the adjusted tasks are then used to run the simulation in operation 164 without the need for compiling the testcase and the method proceeds as described above.

[0030] Figure 5 is a flowchart diagram representing the method operations for minimizing compilation time of a test case during development testing of an integrated circuit in accordance with one embodiment of the invention. The method initiates with operation 180 where a test case is identified. For example, the test case may include a number of macro tasks and individual tasks as discussed with reference to Figure 2.

Here, the test case is formatted as a text file. The method then advances to operation 182 where a verified file associated with the test case is generated. In one embodiment, the file is verified as to syntax and format by a script which reads the test case text file of operation 180. For example, the testinput.txt block of Figure 2 may perform this functionality. The method then proceeds to operation 184 where a sequence of tasks of the file is determined. In one embodiment, the verified file is read line by line to identify the sequence of tasks.

[0031] The method of Figure 5 then moves to operation 186 where hardware description language (HDL) tasks associated with the tasks of the file are requested according to the sequence of the file determined in operation 184. Here, the hardware description language tasks are verilog files that have been pre-compiled and stored. For example, the verilog files may be stored in a library of files such as the tasklist.v with reference to Figure 2. In one embodiment, a translation layer, such as the translation layer with reference to Figure 2, correlates the text based tasks to hardware description language based tasks. The method then moves to decision operation 188 where it is determined if a goal associated with the HDL task has been achieved. Here, a design of the integrated circuit is tested according to a simulation in order to determine if a correct response was achieved when executing the test case. If the goal has been achieved, then the design associated with the applicable test case is accepted. If the goal has not been achieved, then the tasks are adjusted in operation 190if the RTL is acceptable, and operations 180 through 188 are repeated as described above. If the RTL is not acceptable, then the RTL is repaired as discussed with reference to Figure 4. It should be appreciated that the need to compile each test case when it is adjusted is eliminated by the embodiments described herein.

[0032] In summary, the present invention provides a scheme for efficient verification/validation of a IC design. The tasks defining test cases that provide stimulation of a DUT are written as text files. The test cases are each verified in order to identify syntax and format errors. Each of the tasks of the test cases are correlated to a compiled file written in a HDL. A translation layer provides the correlation to the appropriate compiled file from the task written as a text file. Thus, where changes occur to the tasks of the test cases, as is common during the design phase, compilation of the test case is avoided. Additionally, fewer seat licenses are required as a result of minimizing the compiling requirements.

[0033] With the above embodiments in mind, it should be understood that the invention may employ various computer-implemented operations involving data stored in computer systems. These operations include operations requiring physical manipulation of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. Further, the manipulations performed are often referred to in terms, such as producing, identifying, determining, or comparing.

[0034] The above described invention may be practiced with other computer system configurations including hand-held devices, microprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers and the like. The invention may also be practiced in distributing computing environments where tasks are performed by remote processing devices that are linked through a communications network.

[0035] The invention can also be embodied as computer readable code on a computer readable medium. The computer readable medium is any data storage device that can

store data which can be thereafter read by a computer system. The computer readable medium also includes an electromagnetic carrier wave in which the computer code is embodied. Examples of the computer readable medium include hard drives, network attached storage (NAS), read-only memory, random-access memory, CD-ROMs, CD-Rs, CD-RWs, magnetic tapes, and other optical and non-optical data storage devices. The computer readable medium can also be distributed over a network coupled computer system so that the computer readable code is stored and executed in a distributed fashion.

[0036] Although the foregoing invention has been described in some detail for purposes of clarity of understanding, it will be apparent that certain changes and modifications may be practiced within the scope of the appended claims. Accordingly, the present embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope and equivalents of the appended claims. In the claims, elements and/or steps do not imply any particular order of operation, unless explicitly stated in the claims.

What is claimed is: